MOTIVATING STUDENTS FOR MAKING EXERCISES IN AN ONLINE COURSE BY PROVIDING THEM SELF-EVALUATION CAPABILITIES

J. Boydens, J. Peuteman, A. Janssens

KU Leuven (BELGIUM)

Abstract

This paper describes the tools and techniques used in an e-learning course on embedded software development. This course is part of the EOLES (Electronics and Optics e-Learning for Embedded Systems) program aimed at students of the North-African countries where a private internet connection is not always reliable. Originally the project was funded as an EU TEMPUS project but it evolved towards a self-supporting course coordinated by the University of Limoges in France (UNILIM). The project aims to provide a third year bachelor degree program relying exclusively on blended learning, including e-learning and remote laboratories. While the complete EOLES course contains 15 TUs (Technical Units), this paper focusses on one single TU: TU-13, dealing with embedded software development, awarding for 6 ECTUs (European Credit Transfer Units). This TU-13 is steered by the University of Leuven in Belgium (KU Leuven), Faculty of Engineering Technology. TU-13 contains a course on the C programming language, the use of microcontrollers and serial communication but this paper mainly focuses on the C programming course. The course can be followed online using streaming services, but a downloadable equivalent is available for those students who do not have access to a reliable internet connection. A typical programming course requires a lot of scaffolding, and thus special care was taken in providing the customized support while students are both online and offline. An important idea is providing the students the possibility to ask questions to their peers in an online discussion forum while the instructor still manages this forum and validates the answers. If necessary, the instructor posts an accurate reply. Next to this forum, students get automated offline feedback on the exercises they make, based on automated software tests provided with default exercise templates running on their proper laptops. Students are requested to submit their solutions at regular time intervals. Although self-evaluation is available, a formal validation of correctness of their exercises is still necessary. Finally, the paper discusses a correlation between the students passing all self-evaluation exercises (formative evaluations) and their results of a written examination (summative evaluation) held at the end of the course.

Keywords: e-learning course, programming language, virtual classroom, exercises.

1 INTRODUCTION

In 2012 the European Commission granted a TEMPUS project EOLES (Electronics and Optics e-Learning for Embedded Systems) with the University of Limoges as central coordinator. The complete project consortium involved 15 universities. 4 European universities: University of Limoges (UNILIM, France), University of Leuven (KU Leuven, Belgium), School of Engineering of the Polytechnic Institute of PORTO (ISEP, Portugal) and University Polytechnica of Bucharest (UPB, Romania). Each of these institutes had its own experience in the creation of online educational material or remote laboratories [1], [2], [3], [4]. Regarding the North African partners, a selection was made spread across three different countries, willing to commit to the project and rolling out the course in their university. Concerning Algeria, UMAB (Abdelhamid Ibn Badis University) from Mostaganem was co-initiator and also the University of Guelma participated. Additionally, the University of Batna and the University of Adrar were additionally chosen in a geographical spread across Algeria. Concerning Tunisia, VUT (Tunis Virtual University), UK (Kairouan University), US (University of Sfax) and ISET Sousse joined the project. Finally, UCA (Cadi Ayyad University), UAE (Adelmalek Essaâdi University) and USMS (Sultan Moulay Slimane University) were the partners from Morocco.

Each academic year, students enrol through one of the North African partner universities or through the University of Limoges in France. Typically the number of candidate students is larger than the number of students which can be accommodated in the remote laboratories implying a selection of students is made by the program coordinator based on a motivation letter. This way each year a manageable group of 35-40 students enrols for the course which lasts one academic year.

In the following sections, we first discuss the educational system in the Maghreb countries before introducing the EOLES course. In Section 3 and Section 4 we explain the need for a new course and what topics it should contain. Section 5 is dedicated to the C programming language course and how we realise self-evaluation capabilities. In Section 6, we draw some conclusions from practical experiences we already had with the previous editions of the course. Finally, a number of ideas for future work are given in Section 7.

2 THE EDUCATIONAL SYSTEM IN THE MAGHREB COUNTRIES BEFORE THE INTRODUCTION OF EOLES

The main motivation behind the creation of an e-Learning course on electronics and optics for embedded systems is threefold. First, there is an acute need in the joining universities of Maghreb countries to have up to date course material on the development of embedded systems. Secondly training of English language skills of their students and improving the competences of the academic staff when teaching in English is an important challenge. Thirdly, the students in the Maghreb countries already completed a two year bachelor degree in Physics, Electrical, Electronics, Automation, Optics, Telecommunications or equivalent, corresponding to 120 ECTS, and are willing to move professionally towards the field of electronics and optics.

2.1 Embedded Systems

In our current live embedded systems are becoming ubiquitous [5]. Both in home environments as well as in professional environments, the use of programmable electronics is taking a steep rise. For example, a modern car is controlled by numerous small embedded systems, ranging from controllers for powered windows to engine control. Some of these systems are even mission- or safety-critical such as an airbag controller. Gates of public parking places are controlled by embedded systems possibly connected to a pay-per-visit system, and several other examples exist. Even in a medical theatre more and more equipment is controlled by individual embedded systems. It is generally known that for every PC sold on the market, 1000 embedded systems are taken into use. Hence, knowing how to build and program such embedded systems is a prerequisite for current technology experts.

2.2 English as a teaching language

In the universities joining this project the main educational language is either French or Arabic. More and more graduating students are moving abroad or joining international companies in their native country. A thorough knowledge of technical terms in English in the domain concerned is required to be able to work in a world economy as we encounter it today. One of the main goals for this EOLES project was not only to train students on technical topics in English, but also to provide accurate and up to date technical content in English for the academic staff. During the roll-out of the EOLES project many of the project meetings started in English, but gradually the participants switched to French. This experience showed that even within the academic staff steering this project, English is not yet their main communication language. A thorough and explicit English training of the academic staff was necessary and has been organised separately.

3 A NEW COURSE AND AN APPROPRIATE COURSE ORGANIZATION

Prior to the EOLES project, there was no course material available in English on embedded systems in the participating Maghreb universities. Since students are spread over different countries, availability of both course material and laboratories at distance was required. This allows students to follow the course from their home location, even at hours outside the regular teaching hours.

3.1 Recordings

Course material is required to be available at all times, so students are able to study at their own pace. Students can also use different kinds of recordings. By providing them streaming content and downloadable video recordings, students who do not have a reliable internet connection at home can download the videos on their own laptop at their university campuses and watch them later on. Theoretical material for those students was made available in different formats. First of all, slides are provided in a downloadable *.pdf format. Secondly, a streaming service where an instructor explains and elaborates each chapter using the same slides is made. Two downloadable versions, one in a high resolution and

hence becoming larger in size, and one in a lower resolution are made available. Students are advised first to download the slides and walk through these slides one at a time. If they need extra explanation on specific topics, they are directed towards the streaming and downloadable recordings for additional information. This way, students are not forced to look at the entire recordings of all chapters. During the interactive moment, as described in the following Section 3.2, students indicated that they get boring looking at the complete recordings. Using the provided formats, students are able to scroll to the required slide and only listen to the more elaborate explanation they need.

3.2 Interactive moment

Although all information is made available to students by the above mentioned recordings, a weekly interactive moment is still organised. It is essential the students have the ability to stay in contact with their teacher. Such an interactive hour every week motivates students in moving forward on the course. For this interactive moment, Adobe Connect© is used. This allows the teachers to elaborate on topics, explain exercise setups and guide students in studying the theoretical contents. Care was taken not to use this interactive moment as a replacement of the theoretical lecture, but to add an extra way to support students.

Since not all students are able to attend these interactive moments, recordings are made for future reference or delayed watching.

3.3 Moodle

All learning content is made available to students using the Moodle system, an open source course management system [6], [7]. This system not only allows the teacher to concentrate all relevant course information in one location, but it gives him also a central agenda system for the interactive moments. Moreover, it allows the teacher to provide students with an access controlled online discussion forum for extra information retrieval. In reality, experience reveals students are also communicating using social media, which can only be welcomed.

4 THE EOLES PROJECT

4.1 Global project overview

The initial development of course material for the EOLES project was funded by the European Commission under its TEMPUS project call. Its aim was to create a third year bachelor degree in electronics and optics for embedded systems. The topics in this course range from basic communication skills to profound technological courses, where Technical Unit TU13 – “Embedded Systems” is one of the advanced courses. An overview of the technical units, can be found in Table 1:

Table 1. *- Technical Unit List*

Technical Unit	Title	ECTU credits		Technical Unit	Title	ECTU credits
TU01	ICT-Introduction to Virtual Learning Environment	3		TU09	Mathematical and analysis tools for physics 2	3
TU02	Mathematical and analysis tools for physics 1	4		TU10	Signal processing	5
TU03	Communication techniques in English	3		TU11	Instrumentation	4
TU04	Analog electronics	4		TU12	Optics	6
TU05	Digital electronics	4		TU13	Embedded systems	6
TU06	Wave and propagation	6		TU14	Enterprise foundation	3
TU07	Power Electronics	6		TU15	Internship(optional)	3
TU08	Business communication techniques in English	3				

4.2 TU13 – Embedded systems

Only when the students have acquired some technical background from previous technical units, TU13 on embedded systems is scheduled. TU13 focuses on the “C” programming language and a number of hardware related topics. Basic “C” programming is taught, which is the main programming language when considering embedded system programming. Especially when a restricted timeframe is available, the C programming language is commonly used. This reveals why the object oriented language “C++” has not been included in TU13.

An overview of the modules composing TU13 is given in Table 2. Approximately one module per week is scheduled in the interactive moments. Notice however students have the opportunity to move faster forward in the course, since all course materials are available in the Moodle platform at the start of the course.

Table 2. *Overview TU13 Embedded Systems – Module overview*

Module	Description		Module	Description
Module 01	Introduction to C-programming - Part I		Module 04	Interrupts and Timers
Module 02	Introduction to C-programming - Part II		Module 05	Serial communication part I
Module 03	Introduction to microcontrollers		Module 06	Serial communication part II; Miscellaneous

A more detailed overview of the chapters of C-programming in Module 1 and Module 2 is available in Table 3.

Table 3. *Programming course – Chapter overview*

Nr	Description		Nr	Description
01	Introduction		05	Compound Data Types
02	Simple Data Types		06	Programming with Functions
03	Operators		07	Pointers
04	Controlling Program Flow		08	Advanced Pointers

5 THE C PROGRAMMING LANGUAGE

To study the C programming course, it is sufficient to be a novice programmer, the course allows to gradually build up complexity which implies even experienced programmers are challenged. Advanced topics such as *pointers* and *pointer-arithmetic* are included since these principles are regularly used when programming embedded systems.

5.1 The use of an IDE

For this TU13, an Integrated Development Environment (IDE) is used to support students when developing programming code. In real life, depending on the selected hardware, engineers are able to use such an IDE environment or not.

An IDE is a software application which provides comprehensive facilities to computer programmers for software development. It normally consists of a source code editor, a compiler and a debugger. Most modern IDEs also offer intelligent code completion features and a number of *plugins* which add extra features to the environment. For all exercises in Module 1 and Module 2 of “TU13: Embedded Systems” we make use of such an Integrated Development Environment to do most of our programming.

More concretely, Code::Blocks has been used which is a free C, C++ and Fortran IDE built to meet the most demanding needs of its users. Code::Blocks is designed to be very extensible and fully configurable. This approach has the additional advantage it allows an easy installation for the students.

Students are required to solve several exercises which are provided as *workspaces* in Code::Blocks. A workspace is a collection of projects bundled together. This Code::Blocks workspace is compressed in a single zip file per chapter. For example, Fig. 1 gives an overview of the exercises belonging to "Chapter 2 - Simple Data Types". After extracting the zip file, students get the directory structure in Fig. 1.

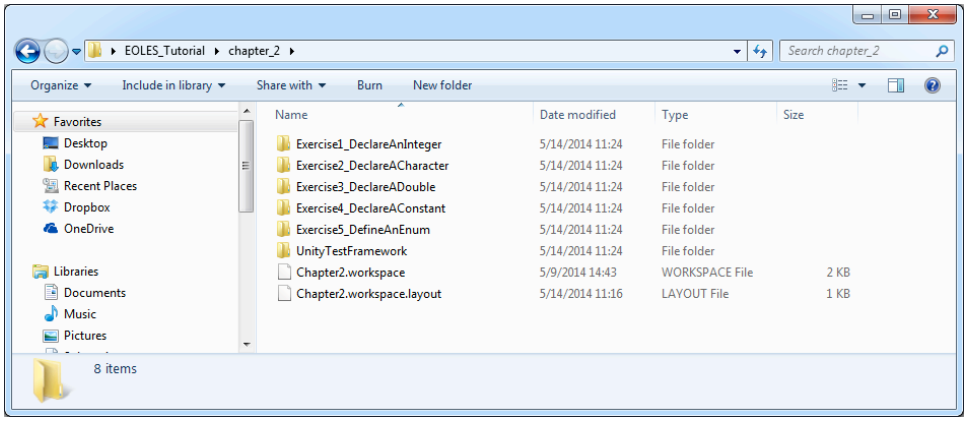

Figure 1 - Directory structure

However, when students open the *workspace* file within the IDE, they obtain an overview of the exercises for the current chapter. As illustrated in Fig. 2, this mimics the directory structure of Fig. 1, but now students work in the controlled environment of the IDE.

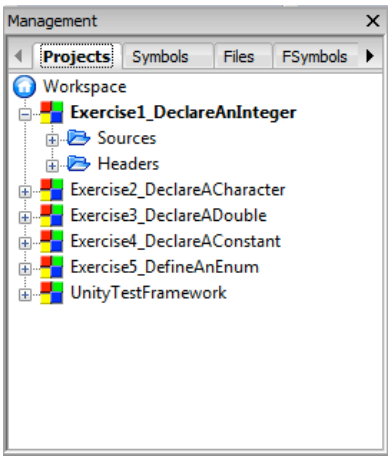

Figure 2 - Workspace in IDE

In the workspace, each exercise has its own project. When "Exercise 1" is expanded, two folders appear: *Sources* and *Headers*. Inside these folders, a number of files are available (main.c, student.c, tests.c and student.h). The main starting point is always "student.c" which is a file containing the instructions for the exercise and additional information.

When students open the file "student.c" (as shown in Fig. 3), he will be required to fill in his name. Finally the exercise instructions are provided.

The file "student.c" always contains areas which have the tags "START OF STUDENT CODE" and "END OF STUDENT CODE" where the student has to add the C programming code. Sometimes, it is also necessary to add code to the "student.h" header file. In such a situation, the required instructions are provided to the student.

To help students, code areas always have TODO comments. These special comments are automatically recognized by Code::Blocks and added to the TODO list shown on the right of the screen (see Fig. 3). This approach helps the students i.e. they know where they have to add their C code.

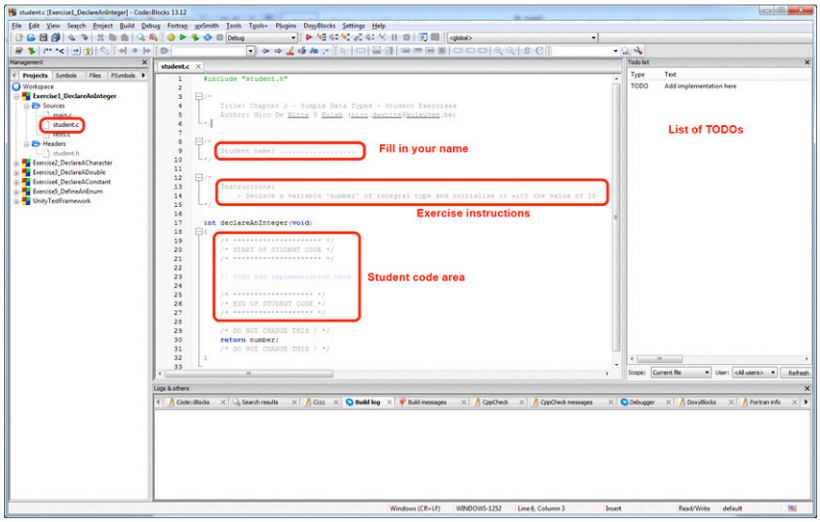

Figure 3 - Contents of file "student.c"

5.2 The use of software tests

All student exercices are accompanied by test code provided by the instructor. Students are not expected to know how the testing framework, operating behind the scene, works. The students merely use the software tests to validate their C implementation. This concept is based on the principle of Test-Driven-Development of software, which has already proven its value in training of novice programmers [8], [9]. Once the students have implemented their solution and run the project, their code will automatically be tested by the instructor's testing code.

This approach gives the students immediate online feedback concerning the correctness of their code. Of course their code is also finally verified by an instructor upon delivery, but this mechanism allows both the student and the instructor to validate the code in an easy and quick way.

In the example of Fig. 4, the code of a novice programmer student has been built and executed. A number of build errors are obtained since the implementation has not been provided yet.

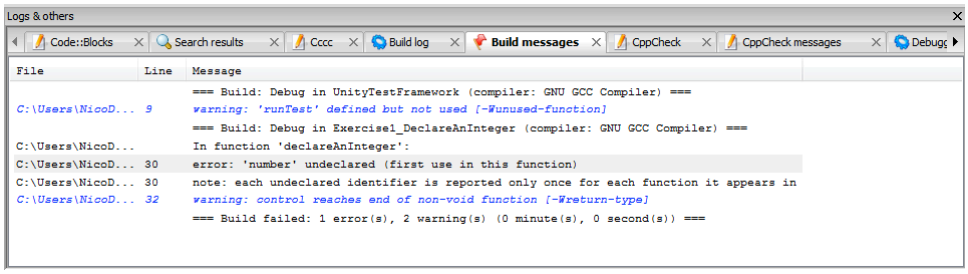

Figure 4 - Build message

This build message does not provide a lot of information to a novice programmer. But the students are motivated to look at the instructions at the top of the file and the TODO-message as illustrated in Fig. 5. In case of Fig. 5, the instructions tell the student to create a variable called *number* of integral type and to initialize it with a value equal to 15. The students have to declare the variable *number* inside the student code area.

Figure 5 - Solving exercise

When the student saves, builds and runs his code, he obtains the output shown in Fig. 6. Here, the results of the automatic tests, carried out on his code, are visualised. Since Fig. 6 deals with a small exercise, it only required a single test. More complicated exercises are covered by multiple tests.

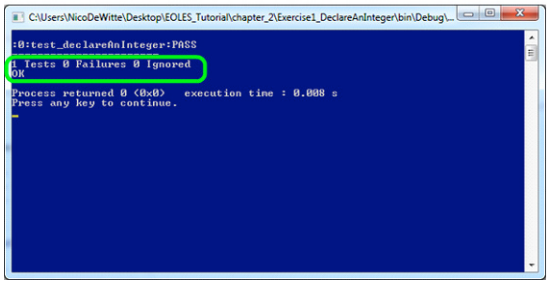

Figure 6 - Sample output

The output of the tests serve as a trouble-shooting guideline for the students' solution. For example, in Fig. 7 a small mistake has been introduced by changing the value of the variable "number" from 15 to 16. When the student saves, builds and re-runs the project, the student sees that the project is compiled without errors but that the test fails as indicated in Fig. 7. The test results even tell him what went wrong (the test *expected 15* but it *received 16*). Finally, the student knows which mistake needs to be fixed. After saving, building and running the project, the final project should pass all tests.

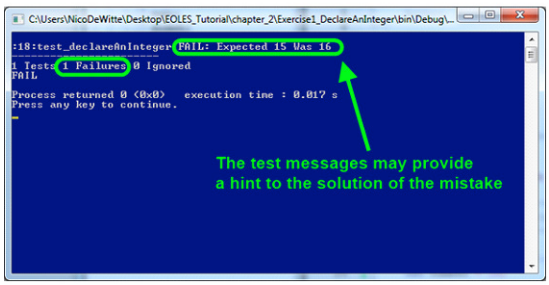

Figure 7 - Failing test

5.3 Support via online forum

As expected by the teaching staff in an early stage of the EOLES project, the students need excessive scaffolding when making programming exercises, especially the novice programmers. The support they obtain through instructions and feedback from the automated tests is very valuable, but from time to time the students still have problems with theoretical topics. To face this consideration, the instructors additionally provide an online forum on the Moodle platform to allow peers (students) to help each other. In Fig. 8, a sample of a student request and the answer by a peer is shown. As illustrated, the peer waits for confirmation of his reply. Finally, at the bottom the acknowledgement of the instructor is shown. Students are motivated to help each other, active cooperation on the forum is rewarded by a bonus point on their final evaluation. Hence, the teachers typically wait one day before posting a reply, allowing the peers to participate in the forum but preventing that the requesting student has to wait too long for a correct answer.

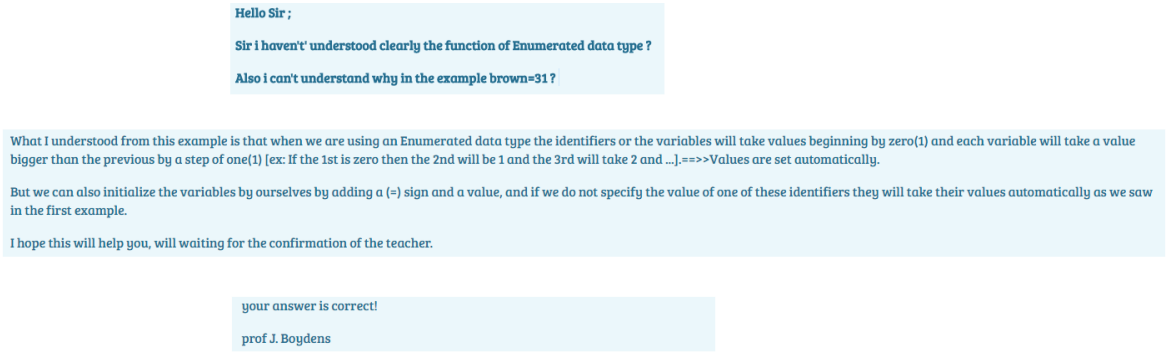

Figure 8 – Forum stimulating student interaction

Continuing in the same online forum discussion thread, a student asks a second question related to a more specialized topic. No peers were keen to provide a correct answer. As a consequence the instructor posts the reply himself so that the thread is not stalled on this topic.

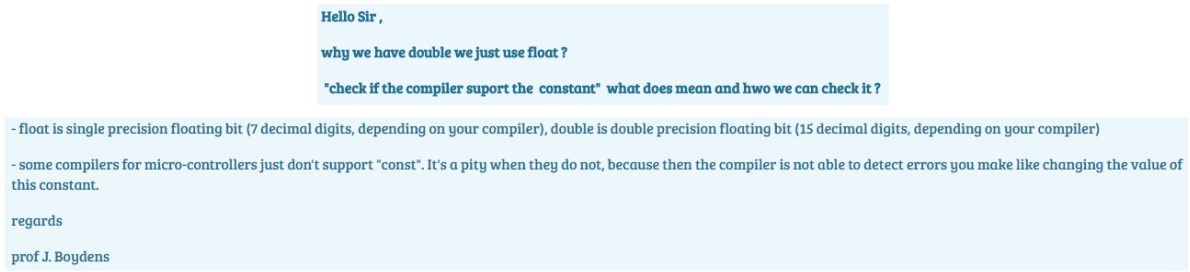

Figure 9 – Forum including an instructor reply

By managing a forum this way, students can rely on the fact that the posted replies are correct. As already mentioned, experience reveals that students also communicate with each other using social media which is a good behaviour. Students trigger their peers when replies are posted and confirmed.

5.4 Formative and summative feedback

In the previous sections, two ways of feedback to the students have been discussed. First, the immediate automated formative feedback in their Integrated Development Environment is provided by means of software testing responses. Secondly, delayed formative feedback is obtained by using a forum accessible to all students in their group and the instructor. The instructor is able to follow the questions coming in and the answers being posted. In this way, the instructor can confirm the correctness of a given answer or he can provide additional information on a number of topics.

Students also get summative feedback through an online intermediate examination performed at home. This examination is scheduled at a pre-defined moment and students should complete it within a timeframe of one hour. The students get a number of multiple choice questions to answer. Both the order of questions and the order of answers are shuffled for each student individually in order to pre-

vent students from cheating. During the entire examination, the webcams of the students are required to be life and a recording is made to provide as much supervision as possible in an online environment. At the end of TU13, an examination on paper in a controlled university environment completes the overall evaluation. The total scores for the students are partitioned in three parts: 25% on the solutions of the programming exercises (validated by re-executing all software tests by the instructor), 25% on the intermediate online examination (performed at home in front of their webcam) and finally 50% on the final examination in the controlled environment of a university.

6 CONCLUSIONS

6.1 Marks obtained by the students

Students making all programming exercises are expected to obtain good overall results. This expectation is confirmed by the intermediate online examination where students have good to very good quotations. Only exceptional students did not submit their exercises in time and even some of these students still got good scores on the intermediate examination. Quotations on exercises in recent years ranged between 0% and 95%, with an average of 60% (0% being exceptional students not submitting any exercises). Whilst quotations on the online intermediate examination ranged between 40% and 100% with an average of 91%.

Contrary with the expectations, when confronted with the final examination containing slightly modified questions in comparison with the intermediate online examination, a majority of the students was unable to give the correct answer implying their final results were significantly worse. A possible and plausible explanation is that students use online information and other information available on their PC when performing the intermediate online examination. This information is not available when performing the final examination in the controlled environment of the university.

6.2 Improving the results of the final evaluation

Although students making all programming exercises themselves are expected to be successful in the intermediate online examination and final examination, this was not always the case. Although there is no way to control whether the solutions of the programming exercises circulated among the students, when scrutinizing the code of the students the instructors concluded that distributing the solutions among the students happens rarely. In the future, the instructors will extend the intermediate online examination by using questions from of a larger pool. In contrast with the current pool of 5 questions which are presented to students in an individually random order, with a random order of possible answer choices, the aim is to have 40 questions in the pool, and the students will need to answer 10 arbitrary selected questions.

The instructors will keep the software testing strategy and the moderated forum unchanged, because it shows to be of great help for students enrolling in the course.

6.3 Experiences of the academic staff

It is important to note that the workload for the instructors was initially huge but pays off in the end. Slides needed to be prepared, recordings had to be made and programming exercises had to be extended containing extra testing features. Fortunately, the course material can be reused every academic year which reduces the workload for future academic years.

Moreover, the instructors are looking for a faster way to validate the submitted programming exercises, since they do not only rely on the automated software tests i.e. the programming exercises are still corrected manually.

7 FUTURE WORK

The instructors plan to extend the number of questions for the online intermediate examination. First, the instructors want a larger number of questions available for each individual student. This approach attempts to solve the problem that some students responded correctly on the intermediate examination but were not able to answer a similar question correctly in the final written examination.

The coordinator of the EOLES project and the instructors assume that some students might use other ways to solve the questions of the intermediate online examination. Since it is not possible to factor

out the social media connection during the intermediate examination, other solutions are needed. A fruitful approach seems to give the students less time to answer the questions avoiding they have time to help each other. Moreover, by having a larger pool to select questions for each student separately, the possibility that different students have the same questions decreases. This also implies they will not be able to help each other.

A second improvement which will be stepwise performed every academic year, is adding extra software tests behind the scenes. In this way, the instructors can extend and improve the formative feedback during the programming exercises. This approach provides the students extra user scenarios for their software under test.

ACKNOWLEDGEMENTS

The EOLES project was funded by the European Commission, under contract number 530466-TEMPUS-1-2012-1-FR-TEMPUS-JPCR.

REFERENCES

[1] A. V. Fidalgo, M. Gericota, D. Barataud, G. Andrieu, R. De Craemer, M. Cristea, A. Benachenhou, M. Ankrim, K. Bouchlaghem en P. Ferreira, „The EOLES project," in 2014 IEEE Global Engineering Education Conference (EDUCON), 2014., pp. 211-216

[2] M. Gericota, A. V. Fidalgo, D. Barataud, G. Andrieu, R. De Craemer, M. Cristea, A. Benachenhou, M. Ankrim, K. Bouchlaghem en P. Ferreira, „EOLES course the first accredited on-line degree course in electronics and optics for embedded systems," in *2015 IEEE Global Engineering Education Conference (EDUCON)*, 2015, pp. 410 – 417

[3] M. Gericota, A. Fidalgo, D. Barataud, G. Andrieu, R. De Craemer, M. Cristea, A. Benachenhou, M. Ankrim, K. Bouchlaghem en P. Ferreira, „combining e-technologies & e-pedagogies to create online undergraduate courses in engineering, an example of a successful experience," in *EDULEARN 16 Proceedings*, Barcelona, 2016, pp. 4209-4218

[4] J. Peuteman, A. Janssens, R. De Craemer, J. Boydens, A. Zabasta en A. Fedotov, „Integration of the European bachelor master degree concept at Belarusian universities for physics and engineering students," in 2016 XXVth International Scientific Conference Electronics (ET), 2016, pp. 47 - 50.

[5] P. Marwedel, Embedded System Design, Dordrecht, Springer Netherlands, 2011.

[6] J. Cole en H. Foster, Using Moodle: Teaching with the Popular Open Source Course Management System, OŔeilly Media, 2007.

[7] A. Janssens, "Active Use of an E-environment", Workshop on ICT environment tools of Erasmus+ project "Physics", Improvement of master level education in the field of physical sciences in Belarusian universities", April 2017, pp. 1- 50

[8] J. Boydens, P. Cordemans, H. Hallez, "On Using Test-Driven Development to Tutor Novice Engineering Students using Self-Assessment", in Proceedings of the 43rd SEFI Annual Conference (29 June - 2 July 2015, Orléans), Hawwash, K., Léger, C. Eds., 2015, pp. 182-190.

[9] K. Beck, Test Driven Development: By Example, Addison-Wesley Professional, 2002